Beeldverwerken Lab 5

Max Bos
10669027

Haischel Dabian
10336230

May 21, 2017

1 Introduction

For this assignment a data set of 550 images from an omnidirectional camera, with the belonging camera position, was given. The task at hand was to find the position of a test image using a Nearest Neighbour approach. The most similar image to the test image was found by obtaining the train image with the lowest Euclidean distance. Principal Component Analysis (PCA) was used to reduce the dimensionality of the image features to improve the performance of the similarity calculations. This report includes experiments to reducing the dimension of each image from 16800 features to a number between 1 and 300.

2 Theory

2.1 Image Correlation

The similarity between two images, I_1 and I_2, can be calculated by computing their image correlation c:

$$c = I_1 \circ I_2 = \frac{1}{K} \sum_{i=1}^{N} \sum_{j=1}^{N} I_1(i,j) I_2(i,j)$$

where K is a normalizing constant, and $\circ$ denotes image correlation. The larger c, the more similar I_1 and I_2.

The sum of squared differences between pixels of two images I_1 and I_2 reduces to a correlation-based similarity measure under the following conditions:

1. Each image contains one object only
2. The objects are images by a fixed camera under weak perspective
3. The images are normalized in size
4. The energy of the pixel values of each images is normalized to 1: $\sum_{i=1}^{N} \sum_{j=1}^{N} I(i,j)^2 = 1$
5. The object is completely visible in all images

Above conditions hold for the provided dataset. The images in the data set seem to all contain one object and all images seem to have been captured by a fixed camera position using the same angle. The images are normalized in size. The object is completely visible in each image, since they all are panoramic views of the same hall.

Since the dot product is the sum of the products of all corresponding elements in two vectors, by representating images as vectors, simply concatenating their pixels, the correlation becomes a dot product. To apply this trick, the images matrices first need to be converted to feature vectors. This can be achieved using MATLAB's `reshape` functionality, since it allows you to reshape a matrix into a vector.

2.2 PCA

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. PCA is used abundantly in all forms of analysis because it is a simple, non-parametric method of extracting relevant information from confusing data sets. PCA provides a procedure for how to reduce a complex data set to a lower dimension to reveal the sometimes hidden, simplified structure that often underlie it. [1]

PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate, called the first principal component, the second greatest variance on the second coordinate, etc. [2]

The procedure for PCA can be summarized as the following steps.

- Organize a data set as an $m \times n$ matrix X, where m is the number of trials and n is the number of features.
- Subtract off the mean for each feature or column x_j of X.
- Calculate the SVD or the eigenvectors of the covariance of X.

The above procedure could also be performed by organizing the data set as an $m \times n$ matrix X, where m is the number of features and n is the number of trials. Then, the mean for each row x_i of X needs to be subtracted of X. This different notation is utilized in an explanation of the PCA technique, by Shlens. The notation in Shlens for X is transposed relative to it. [1]

3 Algorithm

3.1 PCA

The PCA algorithm is implemented as the function called `pca`. The function takes a data matrix X and an integer value d which defines the number of largest magnitude eigenvalues. `pca` returns the PCA eigenvectors, variances and the PCA eigenvectors projected on the data matrix X. Although, the function file `pca.m` is well commented, pseudo-code of the implementation can be seen below.

```
Given:
    X { m by n matrix of input data (m trials, n dimensions)
    d { number of largest magnitude eigenvalues
Returns:
    signals { PC projected on to original data
    PC { each column is a PC (eigenvector)
    V { Mx1 matrix of variances

X = minus_mean(X)
C = covariance(X)
[PC, V] = eigs(X, d)
V = extract_diagonal_values(V)
signals = PC * X
```

3.2 Nearest Neighbour

The Nearest Neighbour functionality is implemented as the function called `nearest_neigbour`. The function takes a dataset of images containing the image matrices and positions, the number of datapoints in the training set, and the number to reduce the dimension of the images to. The function only performs the dimensionality reduction using PCA when the parameter specifying the dimensionality reduction is greater than zero.

```
Given:
    images { a dataset of images with image matrix and position
    train_n { number of datapoints in training set
    d { number of dimensionality reduction
Returns:
    accuracy { the degree to which the correct positions are found

[train_set, test_set] = separate_data_as_matrix(images, train_n)
positions = get_positions(images)

if d > 0
    [Y, PC, V] = pca(train_set, d)
    train_images = PC
    test_image = Y * minus_mean(test_set)

for i in test_set
    test_image = test_set(i)
    min_distance_idx = min(euclidean_distance(train_images, test_image))
    position_distance = euclidean_distance(positions(train_n+i), positions(min_distance_idx))
    if position_distance < 150
        closest_found++

accuracy = closest_found / size(test_set)
```

4 Experiments

The implementation of PCA is initially tested to reduce a data matrix to a dimensionality of 50. Figure 2 depicts the first nine PCA vectors as images. Figure **??** depicts the eigenvalues of first fifty PCA components. The plot shows that the eigenvalue approaches zero when the order of the PCA component increases. Around $d = 50$ still has value, thus $d = 50$ will be maintained as the number of PCA components.

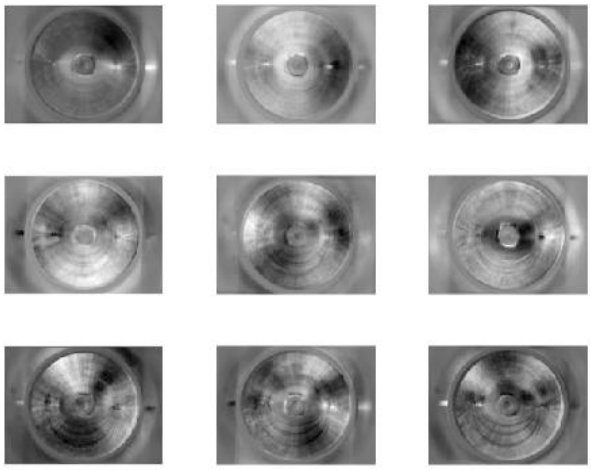

Figure 1: First nine PCA vectors as images, d=50

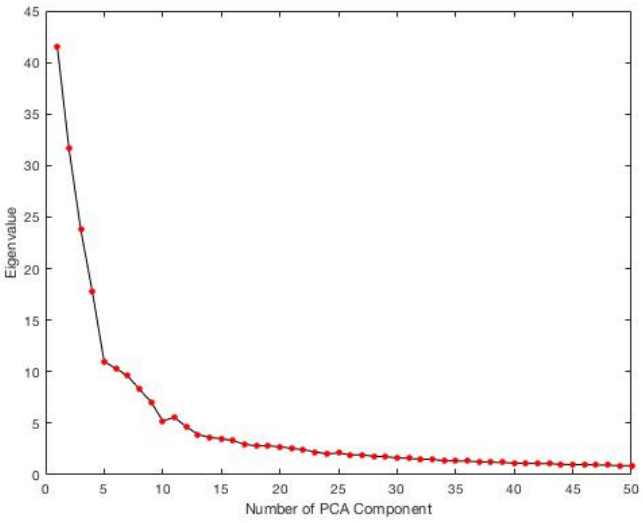

Figure 2: First 50 PCA eigenvalues, d=50

Two tests have been performed to find the training image most similar to a test image. The two tests differ in implementation, where one implementation calculates image differences without performing dimensionality reduction on the data matrix and another where dimensionality reduction is performed on the data matrix. Figure 3 depicts the most similar image found where PCA is performed on the data matrix. The performances of the two different implementation have been timed and a relative speedup of 15 has been obtained using PCA compared to the naive implementation.

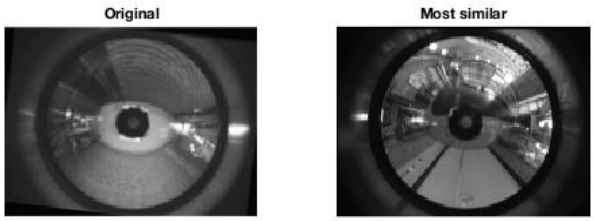

Figure 3: Similar images found using dimensionality reduction

The implementation of the Nearest Neighbour in combination with PCA is used to identify the camera position of all given test images. An experiment has been performed where the dimensionality reduction specified in the PCA ranges from 1 to 300 in increments of 10. Figure 4 depicts the relation between the number of dimensions and the accuracy of the nearest neighbour classification approach. The experiment shows that the highest accuracy of 69.6% is achieved when using around 290 PCA components.

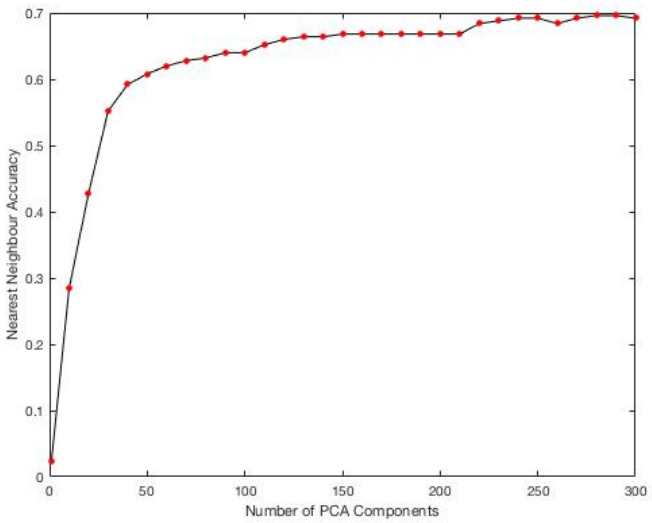

Figure 4: Relation between PCA components and accuracy

The nearest neighbour algorithm has been tested without using PCA, yielding an accuracy of 74.8%. The accuracy from this approach is greater than the highest accuracy when using PCA of around 69.6%.

5 Conclusion

Utilizing PCA to achieve dimensionality reduction in a data set can improve the execution time of a machine learning task. After performing PCA on a provided data set consisting of omnidirectional images, a relative speedup of 15 has been achieved compared to not performing PCA. Although the execution decreased, the highest accuracy yielded when using 290 PCA is 69.6% lower than the 74.8% accuracy yielded when performing the machine learning task on the original data set. This decrease in accuracy may be ascribed to the loss of features when summarizing them in terms of variance.

References

[1] Jonathon Shlens. "A tutorial on principal component analysis". In: *arXiv preprint arXiv:1404.1100* (2014).

[2] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.